SIGSEGV(11) ...

wputman 22 posts since

Aug 16, 2007 Anyone seen this kind of error traceback? I am running the cubed-sphere version of GEOS on Pleiades at Ames with:

%module list

Currently Loaded Modulefiles:

1) nas 2) math/intel_mkl_64_10.0.011 3) comp/intel/10.1.015_64 4) mpi/mpt.1.23.pre

%echo $BASEDIR

/u/dkokron/Baselibs/GMAO-Baselibs-2_1_1-3/Intel-10.1.015_64

I am running a low resolution 1-deg test case. Trying to scale this out I am consistently running into what I think is eventually a memory error with the stack. The stacksize is set to unlimited, and the code is producing bit-wise identical results, with a lower core count run, but as I go above 200+ cores I get this error (same error in different places for different core counts above 200+ cores). It always dies in the same function, but at different time in the code depending on the number of cores I use.

MPI: On host r1i1n8, Program /nobackup/wputman/e520_fp_cubed/scratch/GEOSgcm.x, Rank 12, Process 21854 received signal SIGSEGV(11)

MPI: --------stack traceback-------

MPI: Using host libthread_db library "/lib64/libthread_db.so.1".

MPI: Attaching to program: /proc/21854/exe, process 21854

MPI: Thread debugging using libthread_db enabled

MPI: New Thread 46912515693968 (LWP 21854)

MPI: 0x00002aaaab3127c5 in waitpid () from /lib64/libc.so.6

MPI: (gdb) #0 0x00002aaaab3127c5 in waitpid () from /lib64/libc.so.6

MPI: #1 0x00002aaaaadbd71c in MPI_SGI_stacktraceback ()

MPI: from /nasa/sgi/mpt/1.23.pre/lib64/libmpi.so

MPI: #2 0x00002aaaaadbdd63 in slave_sig_handler ()

MPI: from /nasa/sgi/mpt/1.23.pre/lib64/libmpi.so

MPI: #3 <signal handler called>

MPI: #4 0x0000000000daf38f in geos_utilsmod_mp_qsat1_ ()

MPI: #5 0x0000000000644ebd in geos_lakegridcompmodrun2_mp_lakecore_ ()

MPI: #6 0x00000000006401d4 in geos_lakegridcompmod_mp_run2_ ()

MPI: #7 0x00000000011ccf5e in ESMC_FTable::ESMC_FTableCallVFuncPtr ()

MPI: #8 0x000000000115fa75 in ESMC_FTableCallEntryPointVMHop ()

MPI: #9 0x0000000001190080 in ESMC_VMK::vmk_enter ()

MPI: #10 0x000000000118ae4d in ESMC_VM::ESMC_VMEnter ()

MPI: #11 0x000000000115f3cb in c_esmc_ftablecallentrypointvm_ ()

MPI: #12 0x00000000011cf1ed in esmf_compmod_mp_esmf_compexecute_ ()
MPI: #13 0x000000000115d882 in esmf_gridcompmod_mp_esmf_gridcomprun_ ()
MPI: #14 0x000000000062b621 in geos_surfacegridcompmodrun2_mp_dotype_ ()
MPI: #15 0x0000000000608c11 in geos_surfacegridcompmod_mp_run2_ ()
MPI: #16 0x00000000011ccf5e in ESMC_FTable::ESMC_FTableCallVFuncPtr ()
MPI: #17 0x000000000115fa75 in ESMC_FTableCallEntryPointVMHop ()
MPI: #18 0x0000000001190080 in ESMC_VMK::vmk_enter ()
MPI: #19 0x000000000118ae4d in ESMC_VM::ESMC_VMEnter ()
MPI: #20 0x000000000115f3cb in c_esmc_ftablecallentrypointvm_ ()
MPI: #21 0x00000000011cf1ed in esmf_compmod_mp_esmf_compexecute_ ()
MPI: #22 0x000000000115d882 in esmf_gridcompmod_mp_esmf_gridcomprun_ ()
MPI: #23 0x00000000004432dd in geos_physicsgridcompmod_mp_run_ ()
MPI: #24 0x00000000011ccf5e in ESMC_FTable::ESMC_FTableCallVFuncPtr ()
MPI: #25 0x000000000115fa75 in ESMC_FTableCallEntryPointVMHop ()
MPI: #26 0x0000000001190080 in ESMC_VMK::vmk_enter ()
MPI: #27 0x000000000118ae4d in ESMC_VM::ESMC_VMEnter ()
MPI: #28 0x000000000115f3cb in c_esmc_ftablecallentrypointvm_ ()
MPI: #29 0x00000000011cf1ed in esmf_compmod_mp_esmf_compexecute_ ()
MPI: #30 0x000000000115d882 in esmf_gridcompmod_mp_esmf_gridcomprun_ ()
MPI: #31 0x000000000041de5d in geos_agcmgridcompmod_mp_run_ ()
MPI: #32 0x00000000011ccf5e in ESMC_FTable::ESMC_FTableCallVFuncPtr ()
MPI: #33 0x000000000115fa75 in ESMC_FTableCallEntryPointVMHop ()
MPI: #34 0x0000000001190080 in ESMC_VMK::vmk_enter ()
MPI: #35 0x000000000118ae4d in ESMC_VM::ESMC_VMEnter ()
MPI: #36 0x000000000115f3cb in c_esmc_ftablecallentrypointvm_ ()
MPI: #37 0x00000000011cf1ed in esmf_compmod_mp_esmf_compexecute_ ()
MPI: #38 0x000000000115d882 in esmf_gridcompmod_mp_esmf_gridcomprun_ ()
MPI: #39 0x000000000040bb66 in geos_gcmgridcompmod_mp_run_ ()
MPI: #40 0x00000000011ccf5e in ESMC_FTable::ESMC_FTableCallVFuncPtr ()
MPI: #41 0x000000000115fa75 in ESMC_FTableCallEntryPointVMHop ()
MPI: #42 0x0000000001190080 in ESMC_VMK::vmk_enter ()
MPI: #43 0x000000000118ae4d in ESMC_VM::ESMC_VMEnter ()
MPI: #44 0x000000000115f3cb in c_esmc_ftablecallentrypointvm_ ()
MPI: #45 0x00000000011cf1ed in esmf_compmod_mp_esmf_compexecute_ ()
MPI: #46 0x000000000115d882 in esmf_gridcompmod_mp_esmf_gridcomprun_ ()
MPI: #47 0x0000000000eac0b1 in mapl_capmod_mp_mapl_cap_ ()
MPI: #48 0x000000000040b476 in MAIN__ ()
MPI: #49 0x000000000040b422 in main ()

MPI: (gdb) The program is running. Quit anyway (and detach it)? (y or n) answered Y; input not from terminal

MPI: Detaching from program: /proc/21854/exe, process 21854

MPI: -----stack traceback ends-----

MPI: On host r1i1n8, Program /nobackup/wputman/e520_fp_cubed/scratch/GEOSgcm.x, Rank 12, Process 21854: Dumping core on signal SIGSEGV(11) into directory /nobackup/wputman/e520_fp_cubed/scratch

MPI: MPI_COMM_WORLD rank 12 has terminated without calling MPI_Finalize()

MPI: aborting job

MPI: Received signal 11

Tags: fortran, intel, mpi, geos5

oloso *6 posts since*
Aug 16, 2007 1. **Re: SIGSEGV(11)** Dec 2, 2008 1:38 PM

I haven't run much on pleiades. Of course the fact you are able to run on low core count will suggest that at least your BASELIBS build and the loaded MPI module are consistent. Have you tried another MPI library other than MPT with its corresponding BASELIBS build? But before you try this, have you tried to run with undersubscribing nodes and playing with MPT memory placement env variables like

MPI_DSM_DISTRIBUTE

Use the MPI_DSM_DISTRIBUTE shell variable to ensure that each MPI process will get a physical CPU and memory on the node to which it was assigned.If this environment variable is used without specifying an MPI_DSM_CPULIST variable, it will cause MPI to assign MPI ranks starting at logical CPU 0 and incrementing until all ranks have been placed. Therefore, it is recommended that this variable be used only if running within a cpumemset or on a dedicated system.

MPI_DSM_PPM

The MPI_DSM_PPM shell variable allows you to specify the number of MPI processes to be placed on a node. Memory bandwidth intensive applications can benefit from placing fewer MPI processes on each node of a distributed memory host. On SGI Altix 3000 systems, setting MPI_DSM_PPM to 1 places one MPI process on each node.

as well as some other tunable env variables for Infiniband like

MPI_NUM_QUICKS

Controls the number of other ranks that a rank can receive from over InfiniBand using a short message fast path. This is 8 by default and can be any value between 0 and 32.

MPI_NUM_MEMORY_REGIONS

For zero-copy sends over the InfiniBand interconnect, MPT keeps a cache of application data buffers registered for these transfers. This environmental variable controls the size of the cache. It is 8 by default and can be any value between 0 and 32. If the application rarely reuses data buffers, it may make sense to set this value to 0 to avoid cache trashing.

MPI_CONNECTIONS_THRESHOLD

For very large MPI jobs, the time and resource cost to create a connection between every pair of ranks at job start time may be prodigious. When the number of ranks is at least this value, the MPI library will create InfiniBand connections lazily on a demand basis. The default is 2048 ranks.

MPI_CTRL_RING

When the MPI library is in lazy connection mode (see "MPI_CONNECTIONS_THRESHOLD"), it normally sends connection control messages directly between the involved ranks. This may create excessive load on the administration links between the involved hosts. When the number of ranks is at least the value of this variable, the MPI library will use a throttled control ring between the hosts for this traffic. The default is 8192 ranks.

MPI_IB_PAYLOAD

When the MPI library uses the InfiniBand fabric, it allocates some amount of memory for each message header that it uses for InfiniBand. If the size of data to be sent is not greater than this amount minus 64 bytes for the actual header, the data is inlined with the header. If the size is greater than this value, then the message is sent through remote direct memory access (RDMA) operations. The default is 16384 bytes.

-Hamid

clune *113 posts since*
May 31, 2007 2. **Re: SIGSEGV(11)** Dec 2, 2008 5:33 PM

Ultimately this must be some underlying bug that is exhibiting more benign symptoms on the small core count. Not that this helps track things down. Indeed this may ultimately be yet another example of the weird bug that we have on discover where FP errors cause memory "leaks". I have no idea how to track down this monster with the current tools. In theory totalview replay might offer some help as we take the bad reference back until we see what corrupted it. Even then it would probably be a painful journey.